INFORMATICA

Evaluación

- Primer cuatrimestre (40% de la nota final)
  - Temario: programación (Fortran)
  - Reparto de la nota:
    - 40% dos exámenes programados. Exámenes en el aula
    - 40% entregas de prácticas a través de Moodle. (Cuidado con copiar)
    - 20% ejercicios de clase (mini exámenes sorpresa)

Evaluación

- Segundo cuatrimestre (60% de la nota final)
  - Temario: programación aplicada al cálculo numérico
  - Reparto de la nota:
    - 40% dos exámenes programados. Exámenes en el aula.
    - 40% presentación por grupos en clase. entregas de prácticas en grupo a través de Moodle. (Cuidado con copiar)
    - 20% ejercicios de clase (mini exámenes sorpresa)

- Tipos de datos: Intrínsecos
  - Numéricos: (`integer, real, complex`)
  - Lógicos: (`logical`)
  - Carácter: (`character`)

- Tipos de datos: Derivados
  - Arrays
    - Vectores
    - Matrices
  - Estructuras

Clase de hoy

Antes:

- Dudas práctica jueves pasado:
  - Ejercicio números primos: ¿?
    - Optimizar tiempo
  - Sumatorios: ¿Dudas?

$$S_1 = \sum_{i=1}^{n} i^2 \qquad S_4 = \sum_{i=1}^{n} \sum_{j=i}^{m} (i * j + i - j)$$

$$S_2 = \sum_{i=1}^{n} (i+1)^2 \qquad S_5 = \sum_{i=1}^{n} \sum_{j=i}^{n+m} (i - j)$$

$$S_3 = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{i+j}{i} \qquad S_6 = \sum_{i=1}^{n} \sum_{j=0}^{i} \frac{(i-j)^2}{3}$$

Antes:

- Estilo de programación:

```
program main
implicit none
integer, parameter:: n = 100
integer:: s
integer:: i, j

!---Fin declaracion--------------------

s = 0
do i=1,n
      do j=1,n
            if (i <= j) then
                  s = s + i + j
            endif
      enddo
enddo
end program main
```

Antes:

- Uso de las condicionales: I.- **if… else… end if**

```
if ((n-(n/2)*2) == 0) then
        ...
else
        ...
endif
```

Bien!!

```
if((n-(n/2)*2) == 0) then
        ...
elseif((n-(n/2)*2) /= 0) then
        ...
end if
```

```
if((n-(n/2)*2) == 0) then
        ...
endif
if((n-(n/2)*2) /= 0) then
        ...
end if
```

Regular → Mal

Antes:

- Uso de las condicionales: II.- **if… elseif… else… endif**

```
IF(x > 0) THEN
        WRITE(*,*) '+'
ELSEIF(x == 0) THEN
        WRITE(*,*) '0'
ELSE
        WRITE(*,*) '-'
ENDIF
```

Cálculo número de divisores primos

```
integer:: n,i,ndivisores

write(*,*) 'Escribe un numero'
read(*,*) n

ndivisores= 0

do i=2,n-1
        if(mod(n,i)==0) then! n no es primo
                write(*,*) i,' es un divisor de',n
                ndivisores= ndivisores+1
        endif
enddo

if(ndivisores==0) then
        write(*,*) n,' es primo'
else
        write(*,*) 'Aparte de 1 y el mismo,'
        write(*,*) n,' tiene',ndivisores,' divisores'
endif
```

Arrays: Vectores y Matrices

- Un *array* es una colección de datos, denominados *elementos*, todos ellos del mismo tipo y *kind*, situados en posiciones contiguas de memoria.

- Los *arrays* se clasifican en
  - *unidimensionales* (vectores)
  - *bidimensionales* (matrices)
  - *multidimensionales* (tablas)

Arrays: Vectores y Matrices: Declaración

```
program main

 integer :: U(5)          ! Vector
 real    :: V(10)         ! Vector
 real    :: A(2,3)        ! Matriz 2x3
 real    :: Z(3,2)        ! Matriz 3x2
 real    :: T(0:2,-1:0)   ! Matriz 3x2
 integer :: Ta(2,3,4)     ! Tabla
                          ! 2x3x4

end program main
```

```
program main

    integer, dimension(5)      :: U
    real, dimension(10)        :: V
    real, dimension(2,3)       :: A
    real, dimension(3,2)       :: Z
    real, dimension(0:2,-1:0)  :: T
    integer, dimension(2,3,4)  :: Ta

end program main
```

U:

U(1)	U(2)	U(3)	U(4)	U(5)

A:

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)

T:

T(0,-1)	T(0,0)
T(1,-1)	T(1,0)
T(2,-1)	T(2,0)

Arrays: Vectores y Matrices: Declaración

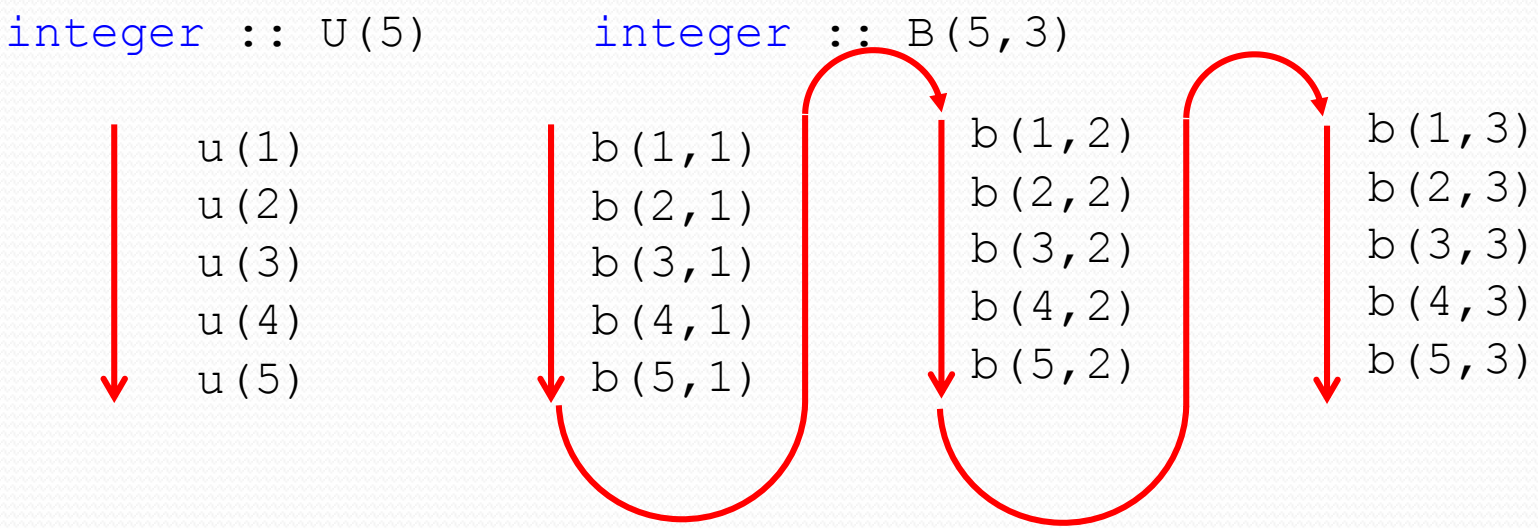
- Lenguaje matemático:

$$A = \begin{pmatrix} 1.1 & 3.8 & 4.3 \\ 1.2 & 5.5 & 7 \end{pmatrix} \qquad T = \begin{pmatrix} 1.1 & 5.5 \\ 1.2 & 4.3 \\ 3.8 & 7 \end{pmatrix}$$

- Lenguaje de programación:

```
A(1,1)=1.1   A(1,2)=3.8   A(1,3)=4.3
A(2,1)=1.2   A(2,2)=5.5   A(2,3)=7.0

T(0,-1)=1.1  T(0,0)=5.5   T(1,-1)=1.2
T(1,0)=4.3   T(2,-1)=3.8  T(2,0)=7.0
```

Vectores y Matrices: Almacenamiento en memoria



- Velocidad de cálculo en grandes códigos

Arrays: Vectores y Matrices: Asignación

```
program asignacion

integer :: U(4)

U = 10

end program asignacion
```

U:

10	10	10	10

```
program asignacion

integer :: U(4)

U = (/0,2,-1,3/)

end program asignacion
```

U:

0	2	-1	3

Arrays: Vectores y Matrices: Asignación

```
program asignacion

integer :: U(4)

U(2) =  0
U(3) = -1
U(1) =  3
U(4) =  2

end program asignacion
```

U:

3	0	-1	2

Arrays: Vectores y Matrices: Asignación

```
program asignacion

integer :: A(2,3)

A = 10

end program asignacion
```

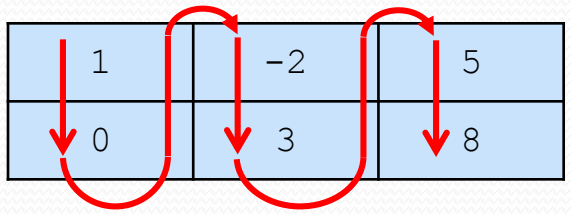
A:

10	10	10
10	10	10

```
program asignacion

integer :: A(2,3)

A = reshape((/1, 0, -2, 3, 5, 8/), (/2,3/))

end program asignacion
```

A:

1	-2	5
0	3	8



OJO!!

Arrays: Vectores y Matrices: Asignación

```
program asignacion

integer :: A(2,3)

A(1,3) = 2
A(1,1) = 10
A(2,1) = -1
A(2,2) = 14
A(1,2) = 3
A(2,3) = -5

end program asignacion
```

A:

10	3	2
-1	14	-5

Arrays: Vectores y Matrices: Declaración II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3
integer, parameter :: Linf = 0
integer, parameter :: Lsup = 10

integer :: U(n)
real :: V(n*m)
real :: A(n,m)
real :: T(Linf:Lsup, Linf:Lsup)

end program declaracion
```

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

end program declaracion
```

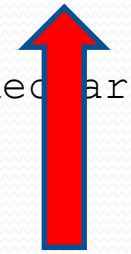
Arrays: Vectores y Matrices: Declaración II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3
integer, parameter :: Linf = 0
integer, parameter :: Lsup = 10

integer :: U(n)
real :: V(n*m)
real :: A(n,m)
real :: T(Linf:Lsup, Linf:Lsup)

end program declaracion
```

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

end program declaracion
```



`U(:)` es un vector y `A(:,:)` una matriz de tamaño desconocidos al principio del programa

Arrays: Vectores y Matrices: Asignación II

- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3

integer :: U(n)
real :: A(n,m)

A = 0.d0
U = 0.d0

end program declaracion
```

- Igual que antes

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

! Cuerpo de programa
n = 2
m = 3
allocate(U(n))
allocate(A(n,m))

A = 0.d0
U = 0.d0

end program declaracion
```

Arrays: Vectores y Matrices: Asignación II

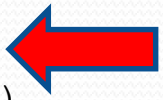
- Parámetro

```
program declaracion

integer, parameter :: n = 2
integer, parameter :: m = 3

integer :: U(n)
real :: A(n,m)

A = 0.d0
U = 0.d0

end program declaracion
```

- Igual que antes

- Dinámica (luego con más detalle)

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

! Cuerpo de programa
n = 2
m = 3
allocate(U(n))
allocate(A(n,m))

A = 0.d0
U = 0.d0

end program declaracion
```



Reserva dinámica de memoria para `U(:)` y `A(:,:)`

Arrays: Vectores y Matrices: Asignación III

- Asignación por bucle

```
program asignacion

integer, parameter :: n = 10
integer :: U(n)
integer :: i

!--- Fin declaracion --------------------

do i=1,n
        U(i) = i*i
end do

write(*,*) U

do i=1,n
        write(*,*) U(i)
end do

end program asignacion
```

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz identidad con n = 5

$$Id = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz identidad con n = 5

$$Id = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
program main

   integer, parameter :: n = 5
   integer :: Id(n,n)
   integer :: i, j

!--- Fin declaracion ------------------

   Id = 0
   do i=1,n
       do j=1,n
           if (i == j) Id(i,j) = 1
       end do
   end do

end program main
```

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij} = \begin{cases} i + j & Si \ \ i \leq j \\ 0 & Si \ \ i > j \end{cases}$$

$$A = \begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij}=\begin{cases} i+j & Si\ i \le j \\ 0 & Si\ i > j \end{cases} \qquad A=\begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion -------------------

do i=1,n
        do j=1,n
                if (i <= j) then
                        A(i,j) = i + j
                else
                        A(i,j) = 0
                end if
        end do
end do
end program main
```

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij}=\begin{cases} i+j & Si\ i \le j \\ 0 & Si\ i > j \end{cases} \qquad A=\begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion -------------------

A = 0
do i=1,n
        do j=1,n
                if (i <= j) A(i,j) = i + j
        end do
end do
end program main
```

Arrays: Vectores y Matrices: Asignación III

Escribir un programa que defina la matriz A_{nxn} con n = 100

$$a_{ij}=\begin{cases} i+j & Si\ i \le j \\ 0 & Si\ i > j \end{cases} \qquad A=\begin{pmatrix} 2 & 3 & 4 & \cdots & 101 \\ 0 & 4 & 5 & \cdots & 102 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 200 \end{pmatrix}$$

```
program main
integer, parameter :: n = 100
integer :: A(n,n)
integer :: i, j
!--- Fin declaracion -------------------

A = 0
do i=1,n
        do j=i,n
                A(i,j) = i + j
        end do
end do
end program main
```

Asignación Dinámica

```
program declaracion

integer :: n
Integer :: m

integer, allocatable :: U(:)
real,    allocatable :: A(:,:)

! Cuerpo de programa

allocate(U(n))
allocate(A(n,m))

A = 0.d0
U = 0.d0

end program declaracion
```

- Inconvenientes de la declaración de un *array* con tamaño fijo:
  - Si el tamaño prefijado es mayor que el número de valores que se van a almacenar, estamos *malgastando* memoria.
  - Si el tamaño prefijado es menor que el número de valores que se van a utilizar, el programa dará un error de ejecución.
- Solución en Fortran: ***arrays dinámicos***.

Asignación Dinámica

```
program declaracion

integer :: n,info
integer, allocatable :: U(:)
! Cuerpo de programa

n = 5

allocate(U(n), stat=info)

if (info > 0) stop '** No hay memoria &
                  & suficiente para U**'

U = 0.d0


deallocate(U, stat=info)

if (info > 0) stop '** U no tenía&
                  & memoria reservada**'


end program declaracion
```

Declaración de U como array dinámico

Reserva de memoria, con control de error
Si la asignación es correcta stat = 0.
En caso contrario stat > 0

Libera la memoria previamente reservada, con control de error

Arrays: Vectores y Matrices: Asignación III

- Trabajo para casa: (subid a Moodle antes del 19 de Octubre)

  - Escribir un programa que reciba por teclado el valor de un número entero n.
  - Calculará la cantidad de números primos menores que n.
  - Creará un vector de tamaño el número de primos donde almacenará dichos primos.
  - Escribe en pantalla el contenido del vector y libera la memoria del mismo.
  - NOTA: se valorará la optimización del código en cuanto al cálculo eficiente de números primos

  - NOMBRE DEL FICHERO: DNI.f95
  - Ejemplo: 12345678R.f95